Data Preprocessing & Cleaning

BigData Week3

2025. 3.20

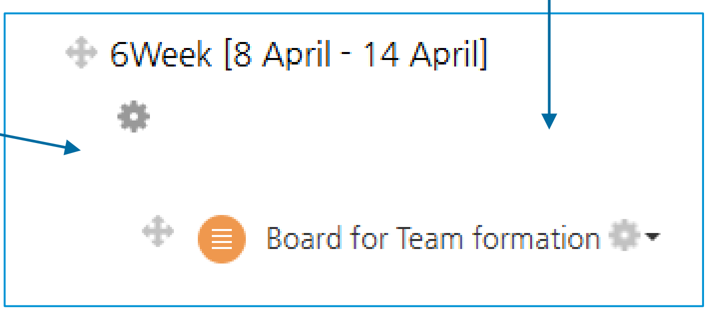
Eunhui Kim (김은희)
ehkim@kisti.re.kr

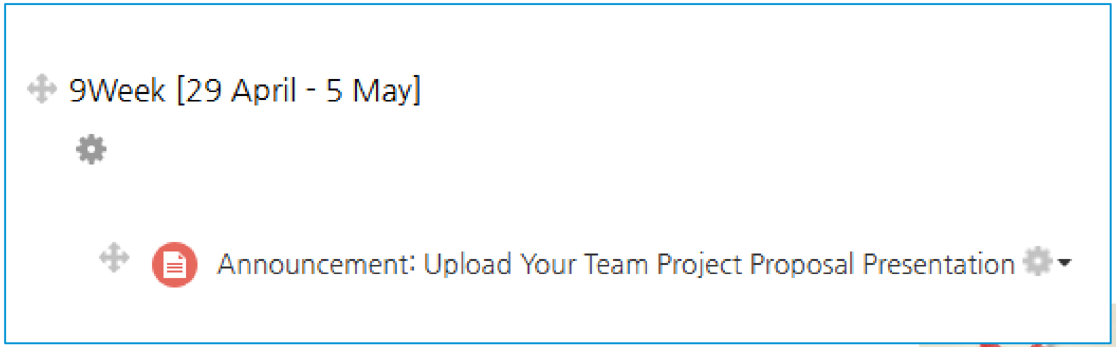
Contents

Notice: Team Project

❖ Team Project Member

- Team composition is limited to 3 or 4 members only
- Submit your team member list through the LMS
  - Submission due : 6th week - April **9** (Wednesday) , 23:55
  - Only one member of a team needs to submit
  - Those who haven't submitted by the deadline will be *randomly assigned*
  - You can utilize the 'Board for Team Formation' board





Review

- Data Analysis with Python
  - Why Python?
    - Beginner-friendly programming language
    - So, people from different disciplines can easily use Python for a variety of different tasks
  - Fundamental Python Libraries
    - Numpy : numerical Python
    - Pandas : Python data analysis
      - Data structures : series (1D array), DataFrame (2D array)
      - Data exploration : head(), tail(), shape, columns, index
      - Data selection (filtering) : condition-based filtering, iloc
      - Aggregate functions : min(), max(), mean(), sum(), count()

Pandas

DataFrame Attributes

df.attribute	description
index	Index labels of the DataFrame
columns	column labels of the DataFrame
dtypes	list the types of the columns
shape	return a tuple representing the dimensionality
values	numpy representation of the data

	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	NaN	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	NaN	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	NaN	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

Pandas

DataFrame Attributes

df.attribute	description
index	Index labels of the DataFrame
columns	column labels of the DataFrame
axes	list the Index labels and column labels

```
df.index
```

```
RangeIndex(start=0, stop=11, step=1)
```

```
df.columns
```

```
Index(['Country', 'Population', 'GDP', 'Continents'], dtype='object')
```

```
df.axes
```

```
[RangeIndex(start=0, stop=11, step=1),
 Index(['Country', 'Population', 'GDP', 'Continents'], dtype='object')]
```

Pandas

DataFrame Attributes

df.attribute	description
shape	return a tuple representing the dimensionality
size	number of elements

```
df.shape
```

```
(11, 4)
```

```
df.size
```

```
44
```

Pandas

DataFrame Attributes

df.attribute	description
values	numpy representation (nparray) of the data

```
df['Population']
```

```
0     32000
1      4000
2      5000
3    135000
4      1000
5     80300
6     12000
7      9000
8      9500
9    120000
Name: Population, dtype: int64
```

```
df['Population'].values
```

```
array([ 32000,   4000,   5000, 135000,   1000,  80300,  12000,   9000,
         9500, 120000], dtype=int64)
```

Pandas

DataFrame Attributes

df.attribute	description
T	Change the rows into columns and columns into rows (transpose)

```
df_t = df.T
df_t
```

	0	1	2	3	4	5	6	7	8	9
Country	CityA	CityB	CityC	CityD	CityE	CityF	CityG	CityH	CityI	CityJ
Population	32000	4000	5000	135000	1000	80300	12000	9000	9500	120000
GDP	80000	45000	10000	9000	62000	35000	55000	12000	73000	81000

Pandas

DataFrame Attributes

df.method()	description
head([n]), tail([n])	show first/last n rows
describe()	generate descriptive statistics
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation
dropna()	drop all the records with missing values

Pandas

❖ DataFrame Attributes

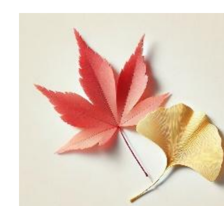
df.method()	description
sample([n])	returns a random sample of the data frame

```python
df.sample(3)
```

	Country	Population	GDP
0	CityA	32000	80000
1	CityB	4000	45000
7	CityH	9000	12000

```python
df.sample(frac=0.2)
```

	Country	Population	GDP
1	CityB	4000	45000
9	CityJ	120000	81000



Pandas

DataFrame Attributes

df.method()	description
sort_value()	sort a DataFrame by the values

```
df.sort_values(by='Population')
```

	Country	Population	GDP
4	CityE	1000	62000
1	CityB	4000	45000
2	CityC	5000	10000
7	CityH	9000	12000
8	CityI	9500	73000
6	CityG	12000	55000
0	CityA	32000	80000
5	CityF	80300	35000
9	CityJ	120000	81000
3	CityD	135000	9000

```
df.sort_values(by='Population', ascending=False)
```

	Country	Population	GDP
3	CityD	135000	9000
9	CityJ	120000	81000
5	CityF	80300	35000
0	CityA	32000	80000
6	CityG	12000	55000
8	CityI	9500	73000
7	CityH	9000	12000
2	CityC	5000	10000
1	CityB	4000	45000
4	CityE	1000	62000

Pandas

DataFrame Attributes

df.method()	description
rename()	rename columns or index of a DataFrame

```
df = df.rename(columns={"Country": "Name"})
```

	Name	Population	GDP
0	CityA	32000	80000
1	CityB	4000	45000
2	CityC	5000	10000
3	CityD	135000	9000
4	CityE	1000	62000
5	CityF	80300	35000
6	CityG	12000	55000
7	CityH	9000	12000
8	CityI	9500	73000
9	CityJ	120000	81000

Data Analysis Process

Gathering data

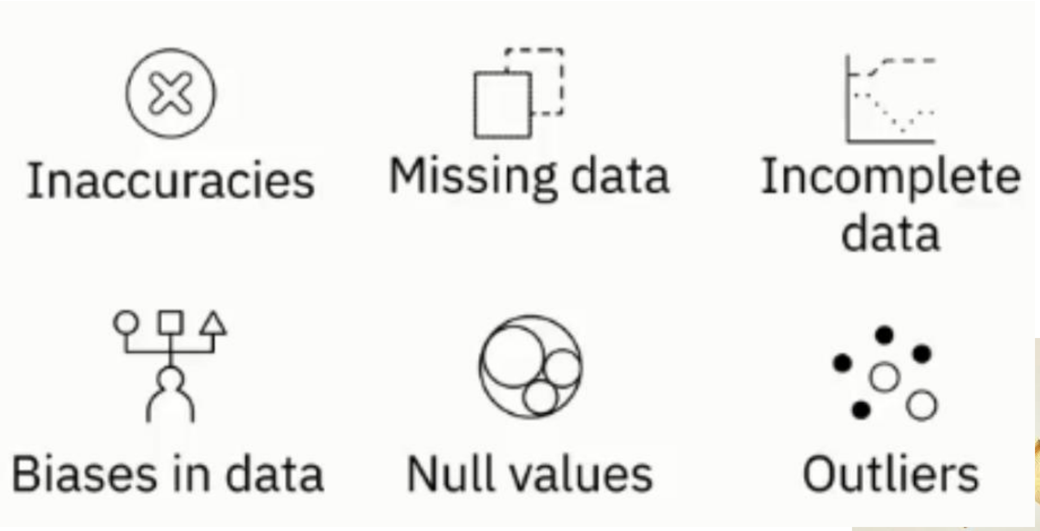
- Identifying possible sources for this data, and best tools for the job

- Data sources
  - Primary data : information obtained directly from the sources
  - Secondary data : information retrieved from existing sources
  - Third-party data : data purchased form aggregators who collect data from various sources and combine it into comprehensive datasets for purpose of selling the data

- Sources for data
  - Databases, web, social media, sensor data, surveys, interviews, observations

Data Analysis Process

- Cleaning data
  - Fixing quality issues in the data and standardizing
  - Raw data needs to get organized, cleaned up, optimized for access, and conform to compliances and standards enforced in the organization
  - Should check:
    - Missing values: drop, replace, of keep the values
    - Data types
      - Type mismatch
      - Compatibility with Python methods
    - Data distribution
    - Data formatting
    - Duplicate data
    - Outliers
    - Syntax errors
    - ...



Data Analysis Process

❖ Analyzing and Mining data

- Extracting, analyzing and manipulating data from different perspectives to understand trends, identify correlations, and find patterns and variations

- Statistical analysis
  - Descriptive statistics: summarize and describe the essential characteristics of a dataset
    - Central tendency (mean, median, mode), dispersion (variance, std), percentiles, skewness, …
  - Inferential statistics: making inferences, predictions, or generalizations about a population based on samples
    - Probability, sampling, hypothesis testing, confidence intervals, regression analysis, …

Data Analysis Process

- Interpreting results
  - Interpreting results, evaluating dependability of analysis and circumstances under which analysis may not hold true

- Presenting your findings
  - Interpreting results, evaluating dependability of analysis and circumstances under which analysis may not hold true

Data Exploration

	A	B	C	D
1	Country	Population	GDP	Continents
2	CityA	32000	80000	Asia
3	CityB	70000	45000	Africa
4	CityC	5000	10000	North America
5	CityD	135000	9000	Asia
6	CityE	1000	62000	Europe
7	CityF	80300	35000	Australia
8	CityG	10000	55000	Europe
9	CityH	9000	12000	South America
10	CityI	9500	73000	Asia
11	CityJ	120000	81000	North America

- Shape

```
df.shape
```

```
(10, 4)
```

- columns

```
df.columns
```

```
Index(['Country', 'Population', 'GDP', 'Continents'], dtype='object')
```

- index

```
df.index
```

```
RangeIndex(start=0, stop=10, step=1)
```

Data Exploration

- Info()
  - Printing information about a DataFrame/Series

```
df.info()
```

	A	B	C	D
1	Country	Population	GDP	Continents
2	CityA	32000	80000	Asia
3	CityB	70000	45000	Africa
4	CityC	5000	10000	North America
5	CityD	135000	9000	Asia
6	CityE	1000	62000	Europe
7	CityF	80300	35000	Australia
8	CityG	10000	55000	Europe
9	CityH	9000	12000	South America
10	CityI	9500	73000	Asia
11	CityJ	120000	81000	North America

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Country     10 non-null     object
 1   Population  10 non-null     int64
 2   GDP         10 non-null     int64
 3   Continents  10 non-null     object
dtypes: int64(2), object(2)
memory usage: 452.0+ bytes
```

```
df.dtypes
```

```
Country       object
Population     int64
GDP            int64
Continents    object
dtype: object
```

Data Exploration

❖ What is Data Type?

- Type of value a variable has and what type of mathematical, relational or logical operations can be applied without causing an error

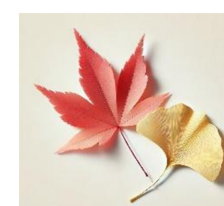
$$5 + 3 \quad = 8$$

$$\text{“5”} + \text{“3”} \quad = \text{“53”}$$

Data Exploration

❖ Data Types (dtypes) in Pandas

Pandas dtype	Usage
object	Text or mixed numeric and non-numeric values
int64	Integer numbers
float64	Floating point numbers
bool	True/False values
datetime64	Date and time values
timedelta	Differences between two datetimes
category	Finite list of text values



Data Exploration

❖ describe()

- Generating descriptive statistics of a DataFrame/Series
- It provides summary statistics for numerical columns by default

```
df.describe()
```

	Population	GDP
count	10.000000	10.000000
mean	47180.000000	46200.000000
std	50590.750143	28654.260882
min	1000.000000	9000.000000
25%	9125.000000	17750.000000
50%	21000.000000	50000.000000
75%	77725.000000	70250.000000
max	135000.000000	81000.000000

Data Exploration

describe()

- .describe(include='O') provides the following statistics for object column

```
df.describe(include='O')
```

	Country	Continents
count	10	10
unique	10	6
top	CityA	Asia
freq	1	3

Data Exploration

- unique()
  - It returns unique values from a specific column

```
df['Continents'].unique()
```

```
array(['Asia', 'Africa', 'North America', 'Europe', 'Australia',
       'South America'], dtype=object)
```

- nunique()
  - Return counts of unique elements

```
df['Continents'].nunique()
```

```
6
```

Data Cleaning

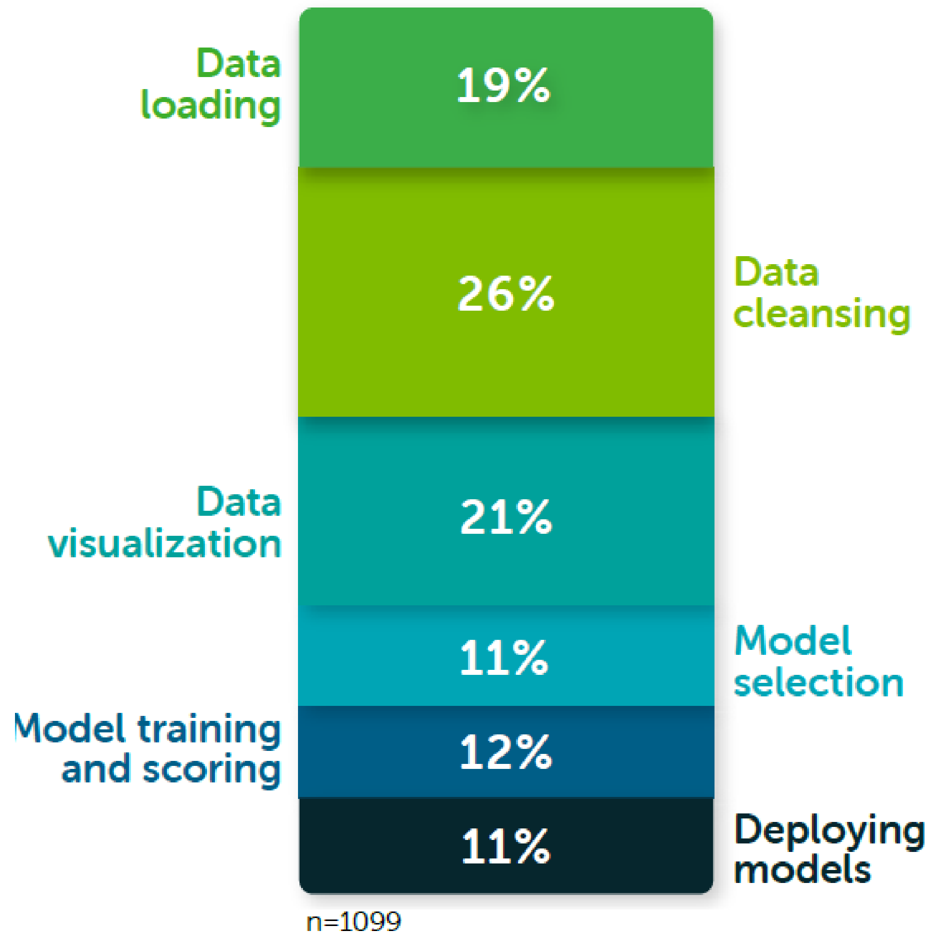
❖ Data Cleaning?

- Process of identifying and correcting errors, inconsistencies, and inaccuracies in datasets
- Primary goal is to prepare data for analysis and modeling by improving data quality

- Data cleaning is essential to:
  - Remove inconsistencies
  - Eliminate errors
  - Handle missing values
  - Standardize formats
  - Enhance data reliability

Data Cleaning

- 2020 State of Data Science
  - How data scientists spend their time:



Data Cleaning

Need for Data Cleaning

- Ideal case:

	A	B	C	D
1	Country	Population	GDP	Continents
2	CityA	32000	80000	Asia
3	CityB	70000	45000	Africa
4	CityC	5000	10000	North America
5	CityD	135000	9000	Asia
6	CityE	1000	62000	Europe
7	CityF	80300	35000	Australia
8	CityG	10000	55000	Europe
9	CityH	9000	12000	South America
10	CityI	9500	73000	Asia
11	CityJ	120000	81000	North America

- What we actually get:

	A	B	C	D
1	Country	Population	GDP	Continents
2	CityA	32000	NA	Asia
3	CityB	70000	45000	Africa
4	CityC	5000		North America
5	CityD	135000	9000	Asia
6	CityE		62000	Europe
7	CityJ	120000	81000	North America
8	CityF	1	35000	Australia
9	CityG	10000	9000	EU
10	CityH	9000	12000	South America
11	CityI	NaN	73000	Asia
12	CityJ	120000	81000	North America

Data Cleaning

- Common issues with Data
  - Missing Value
    - Empty value, NA (Not Available), NULL, NaN (Not a Number)
  - Duplicate Records
    - Multiple identical entries in data
  - Inconsistent Formats
    - Different data, time, or currency formats
  - Outliers/Incorrect Values
    - Extreme values that deviate from the norm
    - Data that contradicts logical constraints

	A	B	C	D
1	Country	Population	GDP	Continents
2	CityA	32000	NA	Asia
3	CityB	70000	45000	Africa
4	CityC	5000		North America
5	CityD	135000	9000	Asia
6	CityE		62000	Europe
7	CityJ	120000	81000	North America
8	CityF	1	35000	Australia
9	CityG	10000	9000	EU
10	CityH	9000	12000	South America
11	CityI	NaN	73000	Asia
12	CityJ	120000	81000	North America

Data Cleaning

Confirming Presence of Missing Value

- Pandas automatically handle missing data represented as "NaN" (short for "Not a Number") during the file reading process

```python
import pandas as pd
```

```python
df = pd.read_csv("country.csv")
df
```

	A	B	C	D
1	Country	Population	GDP	Continents
2	CityA	32000	NA	Asia
3	CityB	70000	45000	Africa
4	CityC	5000		North America
5	CityD	135000	9000	Asia
6	CityE		62000	Europe
7	CityJ	120000	81000	North America
8	CityF	1	35000	Australia
9	CityG	10000	9000	EU
10	CityH	9000	12000	South America
11	CityI	NaN	73000	Asia
12	CityJ	120000	81000	North America

	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	NaN	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	NaN	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	NaN	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

Data Cleaning

- Confirming Presence of Missing Value
  - info() and count()

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Country     11 non-null     object
 1   Population  9 non-null      float64
 2   GDP         9 non-null      float64
 3   Continents  11 non-null     object
dtypes: float64(2), object(2)
memory usage: 484.0+ bytes
```

```
df.count()
```

```
Country       11
Population     9
GDP            9
Continents    11
dtype: int64
```

Data Cleaning

- Confirming Presence of Missing Value
  - isna() (=isnull())
    - Return a boolean indicating if the value is NA

```
df.isna()
```

	Country	Population	GDP	Continents
0	False	False	True	False
1	False	False	False	False
2	False	False	True	False
3	False	False	False	False
4	False	True	False	False
5	False	False	False	False
6	False	False	False	False
7	False	False	False	False
8	False	False	False	False
9	False	True	False	False
10	False	False	False	False

The boolean value
- 'True' is equivalent to the integer value '1'
- 'False' is equivalent to the integer value '0'

```
df.isna().sum()
```

```
Country       0
Population    2
GDP           2
Continents    0
dtype: int64
```

Data Cleaning

Handling Missing Data

- Removing rows/columns with missing values
  - dropna()

```
df_cleaned = df.dropna()
df_cleaned
```

	Country	Population	GDP	Continents
1	CityB	70000.0	45000.0	Africa
3	CityD	135000.0	9000.0	Asia
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
10	CityJ	120000.0	81000.0	North America

```
df_cleaned = df.dropna(axis=1)
df_cleaned
```

	Country	Continents
0	CityA	Asia
1	CityB	Africa
2	CityC	North America
3	CityD	Asia
4	CityE	Europe
5	CityJ	North America
6	CityF	Australia
7	CityG	EU
8	CityH	South America
9	CityI	Asia
10	CityJ	North America

Data Cleaning

- Handling Missing Data
  - Removing rows/columns with missing for specific rows
    - dropna(subset=['column_name'])

```
df = df.dropna(subset=['GDP'])
df
```

	Country	Population	GDP	Continents
1	CityB	70000.0	45000.0	Africa
3	CityD	135000.0	9000.0	Asia
4	CityE	NaN	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	NaN	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

Data Cleaning

❖ Handling Missing Data

- Filling in missing values
  - fillna() : fill or replace missing (NaN) values in a DataFrame with specified values

```
df_fill = df.fillna(100)
df_fill
```

	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	NaN	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	NaN	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	NaN	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

	Country	Population	GDP	Continents
0	CityA	32000.0	100.0	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	100.0	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	100.0	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	100.0	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

Data Cleaning

Handling Missing Data

- Filling in missing values
  - fillna() : fill or replace missing (NaN) values in a DataFrame with specified values

```
df_fill = df.fillna(method='ffill')
df_fill
```

	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	NaN	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	NaN	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	NaN	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	45000.0	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	135000.0	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	9000.0	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

Data Cleaning

Handling Missing Data

- Filling in missing values
  - fillna() : fill or replace missing (NaN) values in a DataFrame with specified values

```
df_fill = df.fillna(method='bfill')
df_fill
```

	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	NaN	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	NaN	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	NaN	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

	Country	Population	GDP	Continents
0	CityA	32000.0	45000.0	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	9000.0	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	120000.0	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	120000.0	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

Data Cleaning

❖ Common issues with Data

- Missing Value
  - Empty value, NA (Not Available), NULL, NaN (Not a Number)
- Duplicate Records
  - Multiple identical entries in data
- Inconsistent Formats
  - Different data, time, or currency formats
- Outliers/Incorrect Values
  - Extreme values that deviate from the norm
  - Data that contradicts logical constraints

	A	B	C	D
1	Country	Population	GDP	Continents
2	CityA	32000	NA	Asia
3	CityB	70000	45000	Africa
4	CityC	5000		North America
5	CityD	135000	9000	Asia
6	CityE		62000	Europe
7	CityJ	120000	81000	North America
8	CityF	1	35000	Australia
9	CityG	10000	9000	EU
10	CityH	9000	12000	South America
11	CityI	NaN	73000	Asia
12	CityJ	120000	81000	North America

Data Cleaning

Identify Duplication

- duplicated()

```
df_dp = df.duplicated()
df_dp
```

```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10      True
dtype: bool
```

```
df_dp = df.duplicated().sum()
df_dp
```


Data Cleaning

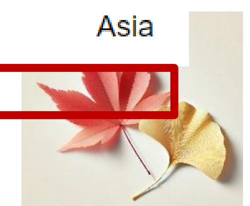
Removing Duplicateds

- drop_duplicates()

```python
df_rev = df.drop_duplicates()
df_rev
```

	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	NaN	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	NaN	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	NaN	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	NaN	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	NaN	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	NaN	73000.0	Asia



Data Cleaning

Common issues with Data

- Missing Value
  - Empty value, NA (Not Available), NULL, NaN (Not a Number)
- Duplicate Records
  - Multiple identical entries in data
- Inconsistent Formats
  - Different data, time, or currency formats
- Outliers/Incorrect Values
  - Extreme values that deviate from the norm
  - Data that contradicts logical constraints

	A	B	C	D
1	Country	Population	GDP	Continents
2	CityA	32000	NA	Asia
3	CityB	70000	45000	Africa
4	CityC	5000		North America
5	CityD	135000	9000	Asia
6	CityE		62000	Europe
7	CityJ	120000	81000	North America
8	CityF	1	35000	Australia
9	CityG	10000	9000	EU
10	CityH	9000	12000	South America
11	CityI	NaN	73000	Asia
12	CityJ	120000	81000	North America

Data Cleaning

Identify Inconsistent Formats

- Checking unique values : describe(), unique(), value_counts()

```
df.describe(include='O')
```

	Country	Continents
count	10	10
unique	10	6
top	CityA	Asia
freq	1	3

```
df['Continents'].unique()
```

```
array(['Asia', 'Africa', 'North America', 'Europe', 'Australia', 'EU',
       'South America'], dtype=object)
```

```
df['Continents'].value_counts()
```

```
Asia             3
North America    3
Africa           1
Europe           1
Australia        1
EU               1
South America    1
Name: Continents, dtype: int64
```

Data Cleaning

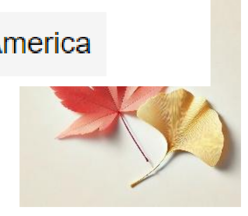
Replacement Values

- replace()

	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	NaN	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	NaN	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	NaN	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

```
df_rp = df.replace('EU', 'Europe')
df_rp
```

	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	NaN	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	NaN	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	Europe
8	CityH	9000.0	12000.0	South America
9	CityI	NaN	73000.0	Asia
10	CityJ	120000.0	81000.0	North America



Data Cleaning

❖ Replacement Values with "str"

- "str" refers to the functionality provided by Pandas to work with string data
- "str" directly access and manipulate string data within the DataFrame

```
df
```

	Name	Phone
0	Jone	123-456-7890
1	Mike	4458879873
2	nick	2759874958
3	Sam	2839405968

```
df['Phone'] = df['Phone'].str.replace('-','')
df
```

	Name	Phone
0	Jone	1234567890
1	Mike	4458879873
2	nick	2759874958
3	Sam	2839405968

Data Cleaning

- Replacement Values with “str”
  - “str” refers to the functionality provided by Pandas to work with string data
  - "str" directly access and manipulate string data within the DataFrame

```
df
```

	Name	Phone
0	Jone	123-456-7890
1	Mike	4458879873
2	nick	2759874958
3	Sam	2839405968

```
df['Name'] = df['Name'].str.upper()
df
```

	Name	Phone
0	JONE	123-456-7890
1	MIKE	4458879873
2	NICK	2759874958
3	SAM	2839405968

```
df['Name'] = df['Name'].str.lower()
df
```

	Name	Phone
0	jone	123-456-7890
1	mike	4458879873
2	nick	2759874958
3	sam	2839405968

Data Cleaning

- Replacement Values with "str"
  - "str" refers to the functionality provided by Pandas to work with string data
  - "str" directly access and manipulate string data within the DataFrame

```
df
```

	Name	Phone
0	Jone	123-456-7890
1	Mike	4458879873
2	nick	2759874958
3	Sam	2839405968

```
df['Name'] = df['Name'].str.capitalize()
df
```

	Name	Phone
0	Jone	123-456-7890
1	Mike	4458879873
2	Nick	2759874958
3	Sam	2839405968

Data Cleaning

❖ Unintended Data Types

```
df['ADD'] = df['Number1']+df['Number2']
df
```

```
df.dtypes
```

```
Number1     object
Number2     object
dtype: object
```

	Number1	Number2
0	1	3
1	2	4
2	3	5
3	4	6
4	5	7

	Number1	Number2	ADD
0	1	3	13
1	2	4	24
2	3	5	35
3	4	6	46
4	5	7	57

Data Cleaning

Unintended Data Types

- astype()

	Number1	Number2
0	1	3
1	2	4
2	3	5
3	4	6
4	5	7

```
df = df.astype('int64')
df.dtypes
```

```
Number1     int64
Number2     int64
dtype: object
```

```
df['ADD'] = df['Number1']+df['Number2']
df
```

	Number1	Number2	ADD
0	1	3	4
1	2	4	6
2	3	5	8
3	4	6	10
4	5	7	12

Data Cleaning

- Common issues with Data
  - Missing Value
    - Empty value, NA (Not Available), NULL, NaN (Not a Number)
  - Duplicate Records
    - Multiple identical entries in data
  - Inconsistent Formats
    - Different data, time, or currency formats
  - Outliers/Incorrect Values
    - Extreme values that deviate from the norm
    - Data that contradicts logical constraints

	A	B	C	D
1	Country	Population	GDP	Continents
2	CityA	32000	NA	Asia
3	CityB	70000	45000	Africa
4	CityC	5000		North America
5	CityD	135000	9000	Asia
6	CityE		62000	Europe
7	CityJ	120000	81000	North America
8	CityF	$\boxed{1}$	35000	Australia
9	CityG	10000	9000	EU
10	CityH	9000	12000	South America
11	CityI	NaN	73000	Asia
12	CityJ	120000	81000	North America

Data Cleaning

Range Limiting

- Condition-based filtering

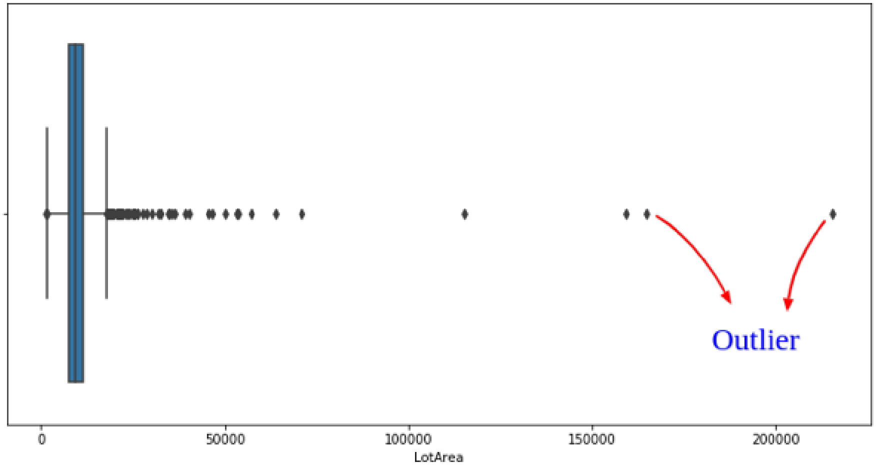
	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	NaN	North America
3	CityD	135000.0	9000.0	Asia
4	CityE	NaN	62000.0	Europe
5	CityJ	120000.0	81000.0	North America
6	CityF	1.0	35000.0	Australia
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
9	CityI	NaN	73000.0	Asia
10	CityJ	120000.0	81000.0	North America

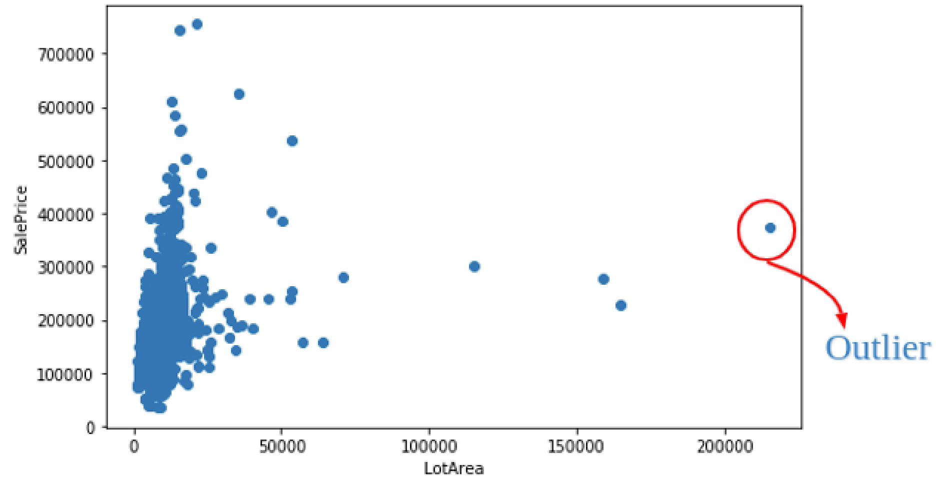
```
df = df[df['Population']>1]
df
```

	Country	Population	GDP	Continents
0	CityA	32000.0	NaN	Asia
1	CityB	70000.0	45000.0	Africa
2	CityC	5000.0	NaN	North America
3	CityD	135000.0	9000.0	Asia
5	CityJ	120000.0	81000.0	North America
7	CityG	10000.0	9000.0	EU
8	CityH	9000.0	12000.0	South America
10	CityJ	120000.0	81000.0	North America

Data Cleaning

- Outlier
  - Data point that significantly deviates from the typical or expected values in a dataset
  - Outliers can be problematic because they can affect the results of an analysis
  - However, they can also be informative about the data you're studying because they can reveal abnormal cases or individuals that have rare traits

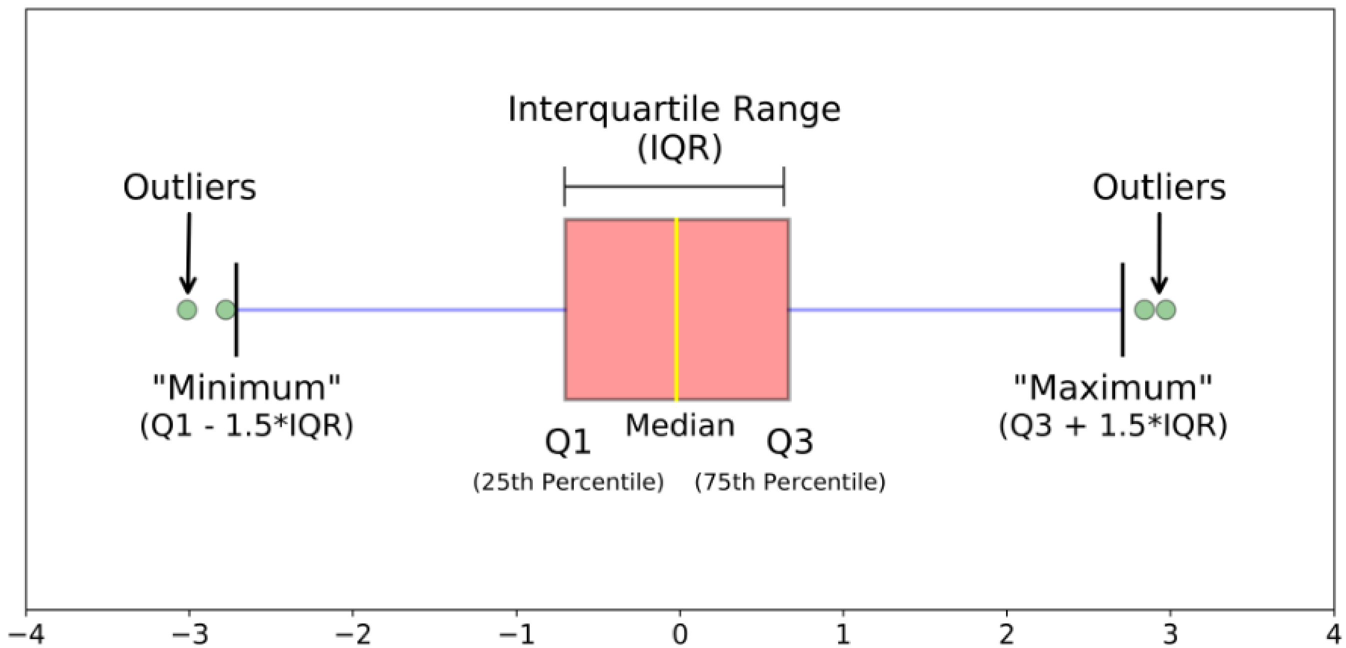




Data Cleaning

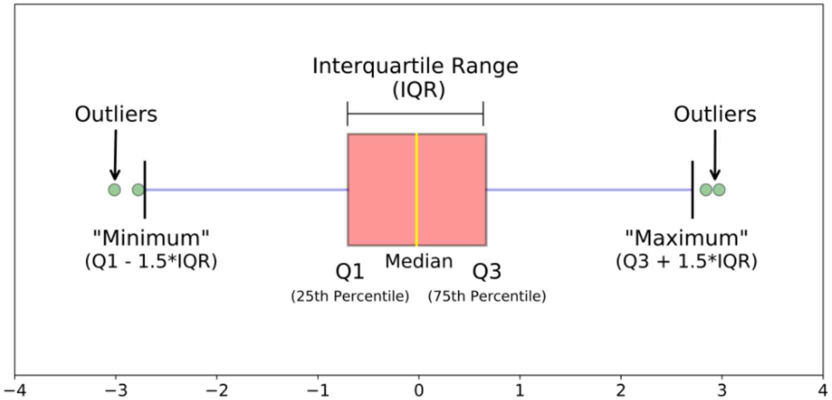
Outlier Detection

- InterQuartile Range (IQR)
  - Statistical measure that assesses the spread or variability of a dataset
  - It is particularly useful in identifying potential outliers
  - Any data point that falls below “Q1-1.5*IQR” or above “Q3+1.5*IQR” is considered an outlier



Data Cleaning

- Outlier Detection
  - InterQuartile Range (IQR)



	Data
0	15
1	20
2	21
3	22
4	22
5	23
6	25
7	26
8	30
9	35
10	40
11	200

```
Q1 = df['Data'].quantile(q=0.25)
Q3 = df['Data'].quantile(q=0.75)
Q1, Q3
```

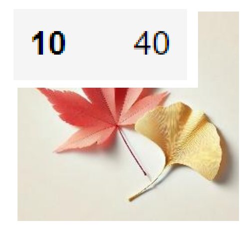
```
(21.75, 31.25)
```

```
IQR = Q3-Q1
IQR
```

```
9.5
```

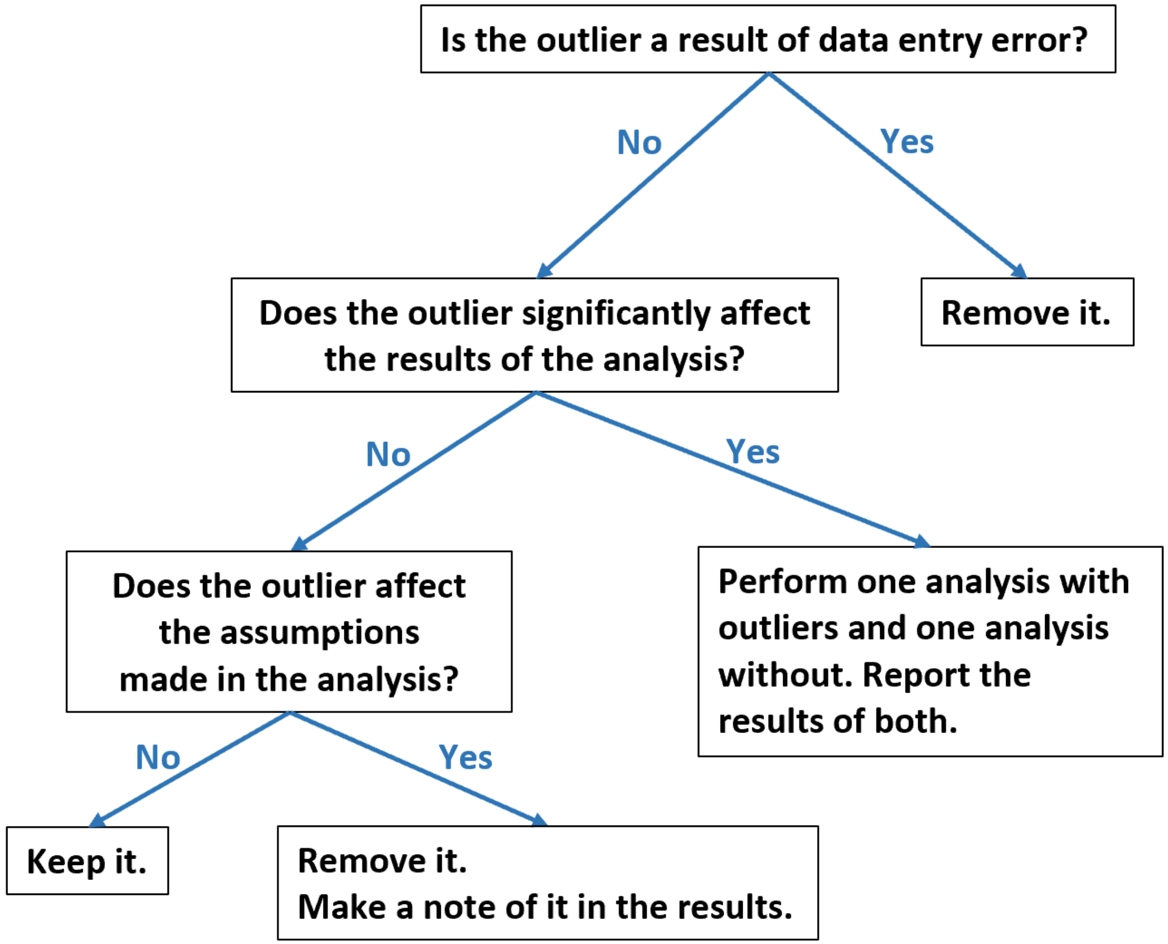
```
IQR_df = df[(df['Data']<=Q3+1.5*IQR) & (df['Data']>=Q1-1.5*IQR)]
IQR_df
```

	Data
0	15
1	20
2	21
3	22
4	22
5	23
6	25
7	26
8	30
9	35
10	40



Data Cleaning

- Outlier Handling
  - Deciding whether to remove or keep outliers



Data Cleaning

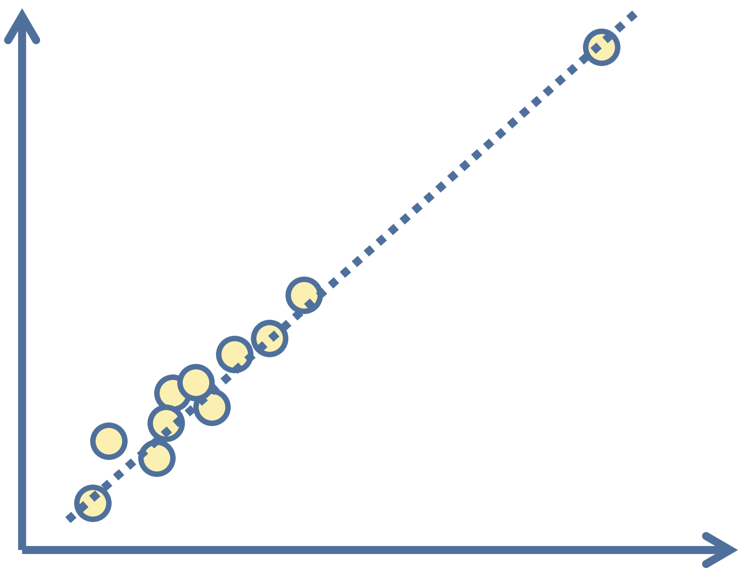
- Outlier Handling
  - Understand the context: Is the outlier a result of data entry error?
    - Consider the domain or field to which the data belongs

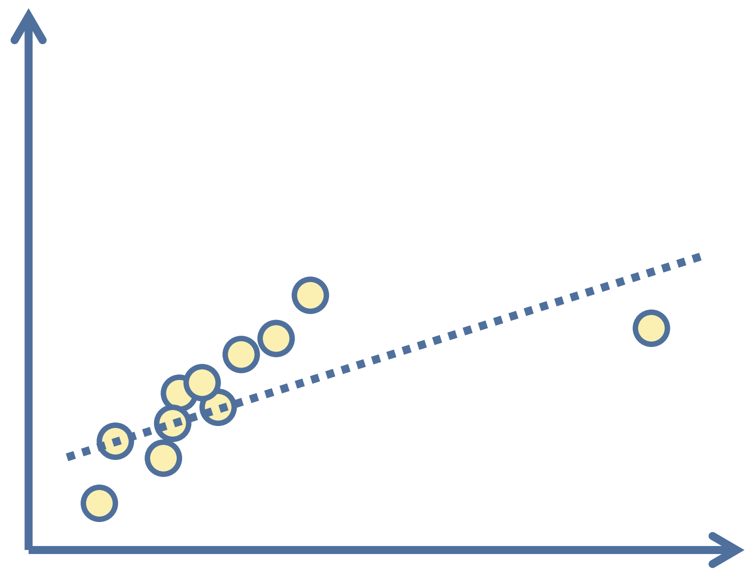
Height
157.0 cm
168.2 cm
175.8 cm
130.1 cm
1800 cm
155.7 cm
182.2 cm
195.1 cm

Data Cleaning

Outlier Handling

- Impact on analysis or model: does the outlier significantly affect the results of the analysis?
  - Assess how outliers affect the analysis or model you plan to perform
  - Do they significantly skew summary statistics or negatively impact model performance?

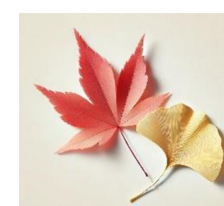




Data Cleaning

Outlier Handling

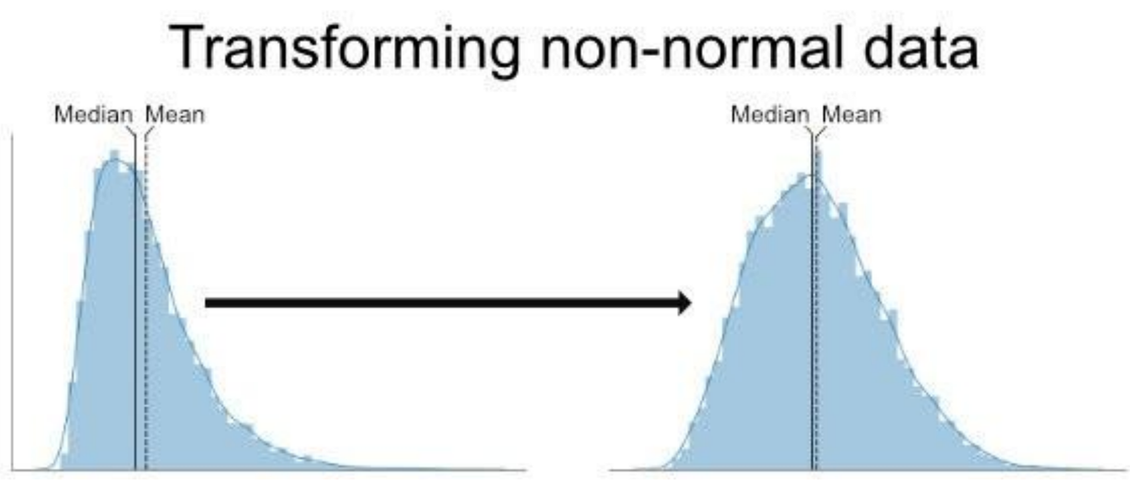
- Does the outlier affect the assumptions made in the analysis?
  - If it does not affect the assumptions, then we can simply keep it in the data
  - However, if it does affect the assumptions remove it
    - removing it from the data and make a note of this when reporting the results



Data Cleaning

❖ Data Transformation

- Modifying the original dataset to make it more suitable for analysis, modeling, or visualization
- It changes relative differences among individual values and consequently also their distribution
- Why?
  - Some statistical analyses and tests require the residuals that are approximately normally distributed and have homogeneous variance
- It helps mitigate the influence of outliers and make data more amenable to statistical analysis and modeling



Data Cleaning

❖ Data Transformation

- There are many types of transformation
  - Log-transformation
  - Square-root transformation
  - Power transformation
  - …

$$y' = \log(ay + c)$$

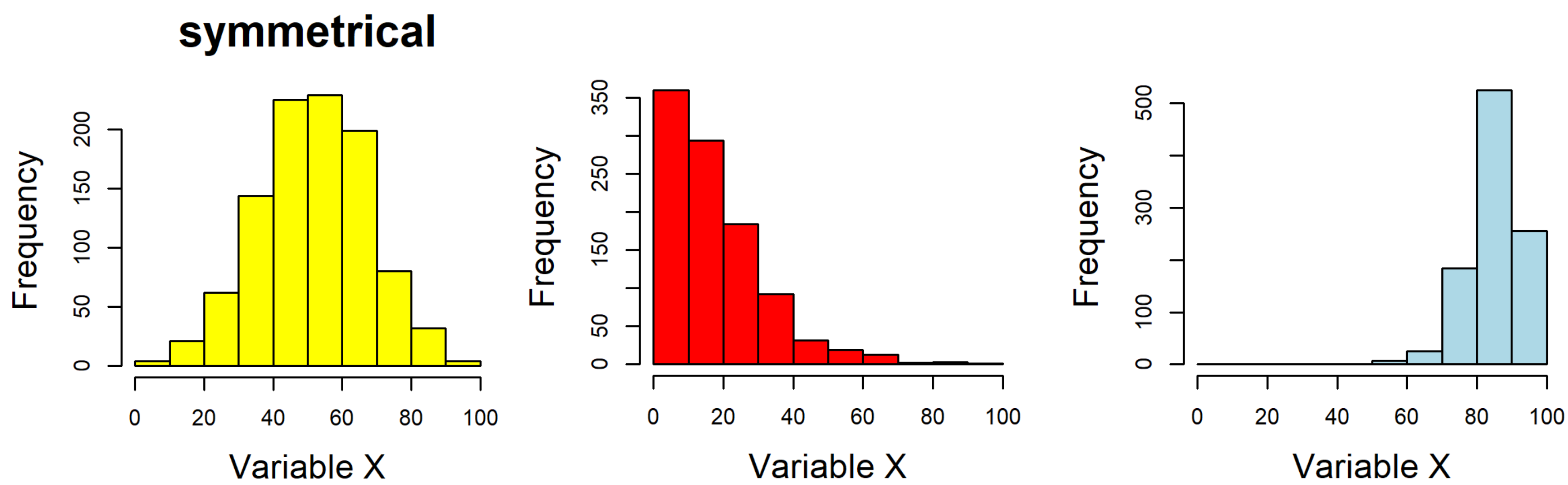
$$y' = \sqrt{y + c}$$

$$y' = y^p$$

Data Cleaning

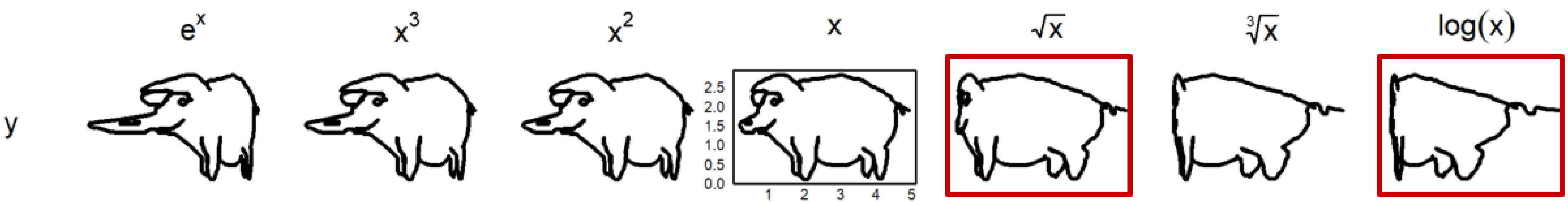
Data Transformation

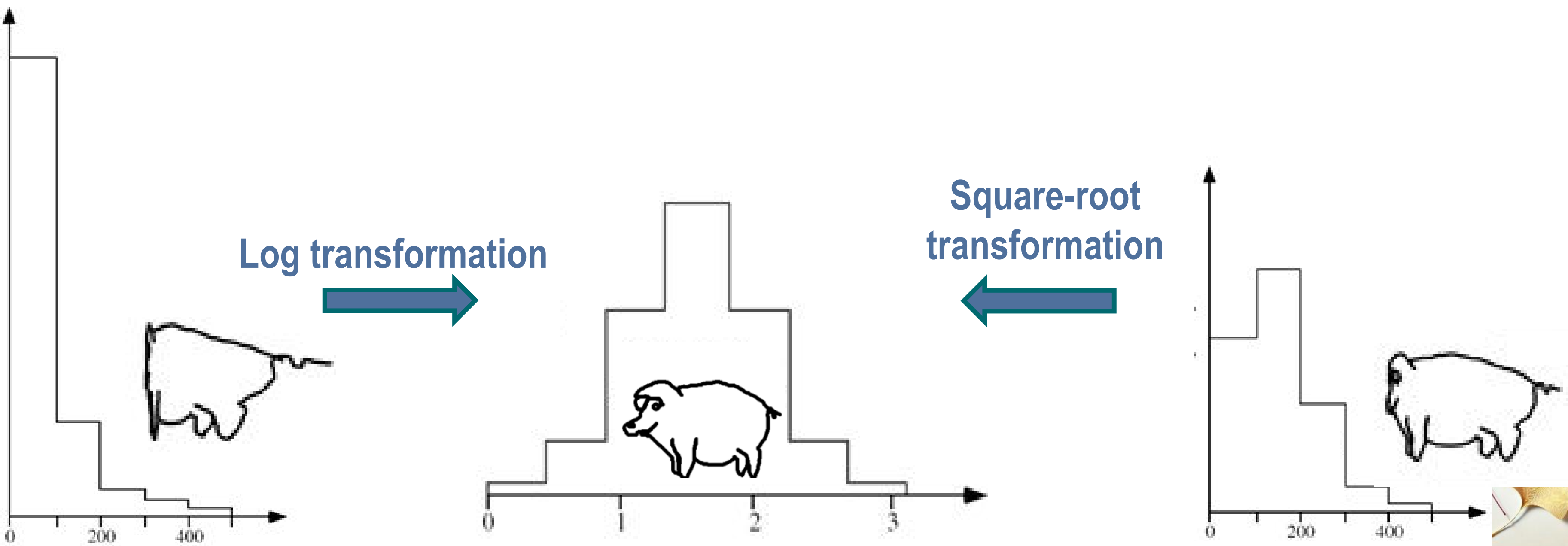
- How to check if our dataset needs transformation?
  - Projecting the values using the histograms and checking whether the distribution is symmetrical, right-skewed or left-skewed
- How can we find the appropriate transformation for our dataset?



Data Cleaning

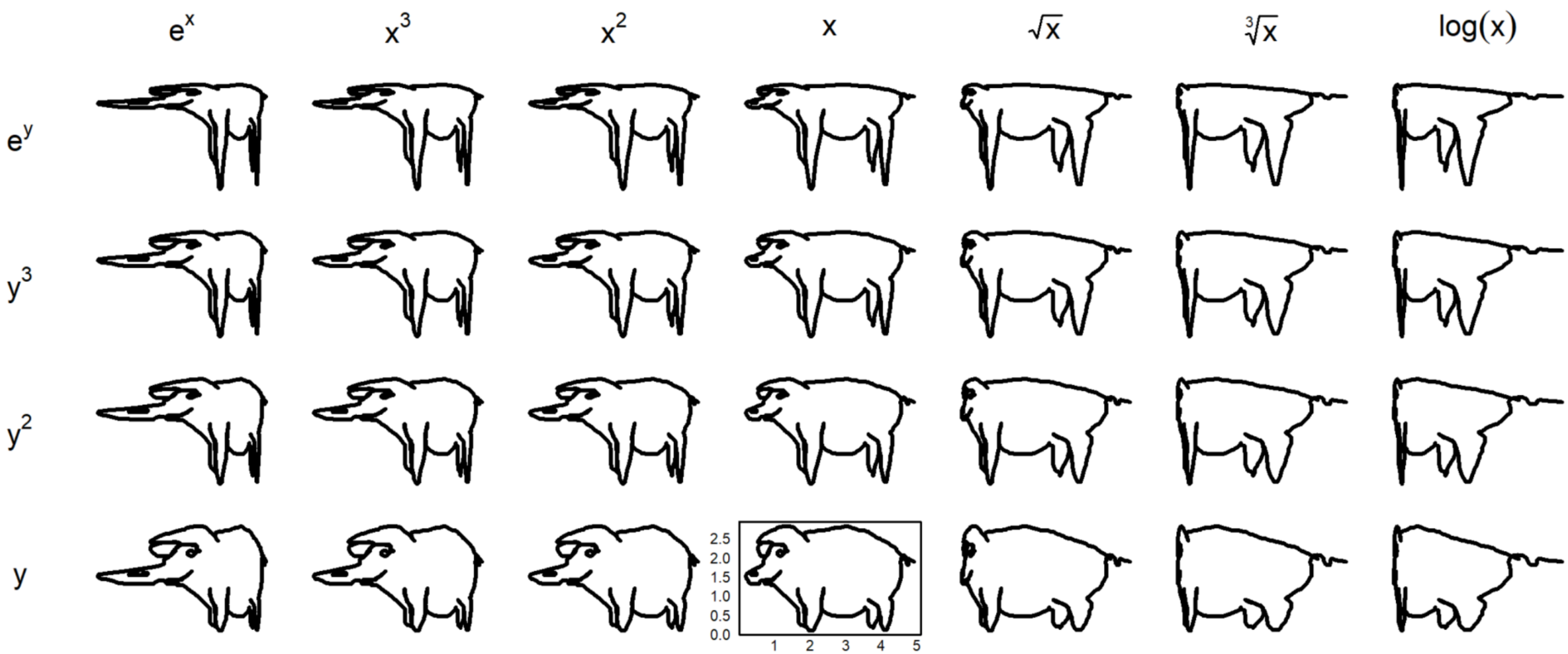
- Data Transformation
  - Transformation to "normal pig"
    - All other pigs can be transformed into the normal pig by the transformation in the upper and left function





Data Cleaning

- Data Transformation
  - Transformation to "normal pig"
    - All other pigs can be transformed into the normal pig by the transformation in the upper and left function



Data Cleaning

Data Standardization

- Changing the data using a statistic calculated from data itself
  - E.g., mean, range, sum of values
- Most common reason to apply is to *remove differences in relative weights (importance)* of individual samples

Suppose we have a dataset of quiz scores with two subjects:

- “Mathematics” with a maximum score of 30
- “English” with a maximum score of 100

→ Normalization : transforms data into a common scale (typically between 0 and 1)

Categorical Variables

- Problem:
  - Most statistical models cannot take in the objects/strings as input
- Solution:
  - Add dummy variables for each unique category and assign 0 or 1 in each category

	Country	Population	GDP	Continents	Population_Category
1	CityB	70000.0	45000.0	Africa	medium
3	CityD	135000.0	9000.0	Asia	high
5	CityJ	120000.0	81000.0	North America	high
6	CityF	1.0	35000.0	Australia	low
7	CityG	10000.0	9000.0	EU	medium
8	CityH	9000.0	12000.0	South America	medium
10	CityJ	120000.0	81000.0	North America	high

"one-hot encoding"

	low	medium	high
1	0	1	0
3	0	0	1
5	0	0	1
6	1	0	0
7	0	1	0
8	0	1	0
10	0	0	1

Categorical Variables

- Use pandas.get_dummies() method
  - Convert categorical variables to dummy variables

```
pd.get_dummies(df['Population_Category'])
```

	Country	Population	GDP	Continents	Population_Category
1	CityB	70000.0	45000.0	Africa	medium
3	CityD	135000.0	9000.0	Asia	high
5	CityJ	120000.0	81000.0	North America	high
6	CityF	1.0	35000.0	Australia	low
7	CityG	10000.0	9000.0	EU	medium
8	CityH	9000.0	12000.0	South America	medium
10	CityJ	120000.0	81000.0	North America	high

	low	medium	high
1	0	1	0
3	0	0	1
5	0	0	1
6	1	0	0
7	0	1	0
8	0	1	0
10	0	0	1

"one-hot encoding"

GroupBy in Pandas

❖ GroupBy method:

- Can be applied on categorical variables
- Powerful tool for performing *group-wise operations*

Team	Score
A	8.1
A	8.3
A	9.2
B	7.1
B	6.5
B	8.6
B	7.3

A	8.1
B	6.5

Min value in each Group

GroupBy in Pandas

- GroupBy method:
  - You start by selecting one or more columns to group your data by and then you can apply various aggregation functions or transformations to each group

① Group the df on the column(s) you want

② Select the field(s) which you want to estimate

	Country	Population	GDP	Continents
0	CityA	32000	20000	Asia
1	CityB	70000	45000	Africa
2	CityC	5000	3000	North America
3	CityD	135000	9000	Asia
4	CityE	50000	62000	Africa
5	CityJ	120000	81000	Asia
6	CityF	3000	35000	EU
7	CityG	10000	9000	EU

```
df.groupby(['Continents'])['GDP'].min()
```

```
Continents
Africa           45000
Asia              9000
EU                9000
North America     3000
Name: GDP, dtype: int64
```

③ Apply aggregation function

GroupBy in Pandas

- GroupBy method:
  - You start by selecting one or more columns to group your data by and then you can apply various aggregation functions or transformations to each group

② Select the field(s) which you want to estimate

```
df.groupby(['Continents'])[['GDP', 'Population']].min()
```

Continents	GDP	Population
Africa	45000	50000
Asia	9000	32000
EU	9000	3000
North America	3000	5000

	Country	Population	GDP	Continents
0	CityA	32000	20000	Asia
1	CityB	70000	45000	Africa
2	CityC	5000	3000	North America
3	CityD	135000	9000	Asia
4	CityE	50000	62000	Africa
5	CityJ	120000	81000	Asia
6	CityF	3000	35000	EU
7	CityG	10000	9000	EU

GroupBy in Pandas

- GroupBy method:
  - You start by selecting one or more columns to group your data by and then you can apply various aggregation functions or transformations to each group

① Group the df on the column(s) you want

```
df.groupby(['Continents','Population_Category'])['GDP'].mean()
```

	Country	Population	GDP	Continents	Population_Category
0	CityA	32000	20000	Asia	medium
1	CityB	70000	45000	Africa	medium
2	CityC	5000	3000	North America	low
3	CityD	135000	9000	Asia	high
4	CityE	50000	62000	Africa	medium
5	CityJ	120000	81000	Asia	high
6	CityF	3000	35000	EU	low
7	CityG	10000	9000	EU	medium

		GDP
Continents	**Population_Category**	
Africa	**low**	NaN
	medium	53500.0
	high	NaN
Asia	**low**	20000.0
	medium	NaN
	high	45000.0
EU	**low**	22000.0
	medium	NaN
	high	NaN
North America	**low**	3000.0
	medium	NaN
	high	NaN

GroupBy in Pandas

- GroupBy method:
  - You start by selecting one or more columns to group your data by and then you can apply various aggregation functions or transformations to each group

```
df.groupby(['Continents','Population_Category'])['GDP'].mean()
```

```
df.groupby(['Continents','Population_Category'])['GDP'].mean().unstack()
```

Population_Category	low	medium	high
Continents			
Africa	NaN	53500.0	NaN
Asia	20000.0	NaN	45000.0
EU	22000.0	NaN	NaN
North America	3000.0	NaN	NaN

Assignment 2

- Submission due : March. 27 th, 23:55
- What to submit : Notebook file (.ipynb)
  - Colab : [File]-[Download]-[Download .ipynb]
  - Kaggle : [File]-[Download Notebook]

IMPORTANT

- Be sure to download the dataset from Assignment-2
  - The file name is "titanic_rev.csv"
  - This is a modified data different from Assignment-1
- Make sure your results are the same as the output presented on this slide
  - Problem 5-7 : depending on whether you accumulate the result into a single DataFrame, the result may differ

Assignment 2

Titanic - Machine Learning from Disaster

Start here! Predict survival on the Titanic and get familiar with ML basics

Assignment 2

- Titanic dataset includes:
  - Passenger ID
  - Passenger Class (1st, 2nd, or 3rd class)
  - Name
  - Sex
  - Age
  - Sibling/Spouse Aboard (SibSp)
  - Parent/Child Aboard (Parch)
  - Ticket Number
  - Fare
  - Cabin Number
  - Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)
  - Whether the passenger survived (1 for survived, 0 for did not survive)

Assignment 2

- Titanic dataset includes:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Passenger	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, M	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, I	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen,	female	26	0	0	STON/O2.	7.925		S
5	4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy,	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Palsson, M	male	2	3	1	349909	21.075		S
10	9	1	3	Johnson, M	female	27	0	2	347742	11.1333		S
11	10	1	2	Nasser, Mr	female	14	1	0	237736	30.0708		C
12	11	1	3	Sandstrom	female	4	1	1	PP 9549	16.7	G6	S
13	12	1	1	Bonnell, M	female	58	0	0	113783	26.55	C103	S

⋮

Assignment 2
Cleaning Titanic Dataset by Pandas

① Problem 1: Load the Titanic dataset from file – Using the Titanic dataset (titanic_rev.csv) that is uploaded on the LMS.

Print the dimension of the dataset.

```
(891, 12)
```

or

891 rows × 12 columns

Assignment 2

Cleaning Titanic Dataset by Pandas

② Problem 2: Print how many non-null values there are in each column

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          712 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Assignment 2
Cleaning Titanic Dataset by Pandas

③ Problem 3: Replace the NA value in "Age" column with the *mean* of "Age". Then, print the *first five* rows.

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	29.741812	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	Female	35.000000	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	NaN	S

Assignment 2
Cleaning Titanic Dataset by Pandas

④ Problem 4: Remove the ‘Cabin’ column. Then, print the column labels. Save the df column with removing Cabin for next problem.

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Embarked'],
      dtype='object')
```

Assignment 2
Cleaning Titanic Dataset by Pandas

⑤ Problem 5: Remove the rows that have a NA value in the "Embarked" column. Then, print the dimensionality of the DataFrame.

```
(889, 11)
```

Assignment 2
Cleaning Titanic Dataset by Pandas

⑥ Problem 6: Print the unique values of 'Sex' Column first. Then, change the value format of the 'Sex' column to use only 'female' or 'male'. Then print the count of unique values in the 'Sex' column.

```
array(['male', 'female', 'Female', 'M', 'F', 'Male'], dtype=object)
```

	count
Sex	
male	578
female	311

dtype: int64

Assignment 2
Cleaning Titanic Dataset by Pandas

⑦ Problem 7: Find outliers in the “Fare” column using the InterQuartile Range (IQR) method. At first print Q1, Q3 and IQR of “Fare” columns. And then print only the rows corresponding to the outliers.

Another answer:
Q1: 7.91, Q3: 31.27
IQR: 23.3646

```
Q1:7.8958, Q3: 31.0
IQR: 23.1042
```

	PassengerId	Survived	Pclass	Name	Sex						
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.000000	3	2	19950	263.0000	S
31	32	1	1	Spencer, Mrs. William Augustus (Marie Eugenie)	female	29.741812	1	0	PC 17569	146.5208	C
34	35	0	1	Meyer, Mr. Edgar Joseph	male	28.000000	1	0	PC 17604	82.1708	C
52	53	1	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	female	49.000000	1	0	PC 17572	76.7292	C
...	...	...	...	...	...	...	...	...	...	...	...
846	847	0	3	Sage, Mr. Douglas Bullen	male	29.741812	8	2	CA. 2343	69.5500	S
849	850	1	1	Goldenberg, Mrs. Samuel L (Edwiga Grabowska)	female	29.741812	1	0	17453	89.1042	C
856	857	1	1	Wick, Mrs. George Dennick (Mary Hitchcock)	female	45.000000	1	1	36928	164.8667	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	29.741812	8	2	CA. 2343	69.5500	S
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.000000	0	1	11767	83.1583	C

116 rows × 11 columns

Q & A

김은희 (ehkim@kisti.re.kr)
TA: Yesim Selcuk (yesimselcuk@kisti.re.kr)